Contents & Goals

Last Lecture:
- RDC in discrete time
- Started: Satisfiability and realisability from 0 is decidable for RDC in discrete time

This Lecture:
- Educational Objectives: Capabilities for following tasks/questions.
  - Facts: (un)decidability properties of DC in discrete/continuous time.
  - What's the idea of the considered (un)decidability proofs?
- Content:
  - Complete: Satisfiability and realisability from 0 is decidable for RDC in discrete time
  - Undecidable problems of DC in continuous time
RDC in Discrete Time Cont’d
Theorem 3.6.
The satisfiability problem for RDC with discrete time is decidable.

Theorem 3.9.
The realisability problem for RDC with discrete time is decidable.
Recall: Proof Sketch

\[ R^D \text{ formula } F, \text{ DT1 } \Rightarrow \]

\[ F \equiv I, [0, n] \]

\[ \Rightarrow \text{ construct } \uparrow \text{ prove } \]

\[ L(F) \subseteq w \in \Sigma^* \text{ word} \]
Sketch: Proof of Theorem 3.6

- give a procedure to construct, given a formula $F$, a *regular* language $\mathcal{L}(F)$ such that

$$\mathcal{I}, [0, n] \models F \text{ if and only if } w \in \mathcal{L}(F)$$

where word $w$ describes $\mathcal{I}$ on $[0, n]$

(suitability of the procedure: Lemma 3.4)

- then $F$ is satisfiable in discrete time if and only if $\mathcal{L}(F)$ is not empty (Lemma 3.5)

- Theorem 3.6 follows because
  - $\mathcal{L}(F)$ can *effectively* be constructed,
  - the emptiness problem is *decidable* for regular languages.
**Construction of $\mathcal{L}(F)$**

- **Idea:**
  - alphabet $\Sigma(F)$ consists of basic conjuncts of the state variables in $F$,
  - a letter corresponds to an interpretation on an interval of length 1,
  - a word of length $n$ describes an interpretation on interval $[0, n]$.

- **Example:** Assume $F$ contains exactly state variables $X, Y, Z$, then

$$
\Sigma(F) = \{ X \land Y \land Z, X \land Y \land \neg Z, X \land \neg Y \land Z, X \land \neg Y \land \neg Z, \\
\neg X \land Y \land Z, \neg X \land Y \land \neg Z, \neg X \land \neg Y \land Z, \neg X \land \neg Y \land \neg Z \}.
$$
Construction of $\mathcal{L}(F)$ more Formally

**Definition 3.2.** A word $w = a_1 \ldots a_n \in \Sigma(F)^*$ with $n \geq 0$ describes a **discrete** interpretation $\mathcal{I}$ on $[0, n]$ if and only if

$$\forall j \in \{1, \ldots, n\} \quad \forall t \in ]j-1, j[ : \mathcal{I}[a_j](t) = 1.$$ 

For $n = 0$ we put $w = \varepsilon$.

- Each state assertion $P$ can be transformed into an equivalent **disjunctive normal form** $\bigvee_{i=1}^{m} a_i$ with $a_i \in \Sigma(F)$.
- Set $DNF(P) := \{a_1, \ldots, a_m\} \subseteq \Sigma(F))$.
- Define $\mathcal{L}(F)$ inductively:

  $$\mathcal{L}([P]) = DNF(P)^+,$$
  $$\mathcal{L}(\neg F_1) = \Sigma(F)^* \setminus \mathcal{L}(F_1),$$
  $$\mathcal{L}(F_1 \lor F_2) = \mathcal{L}(F_1) \cup \mathcal{L}(F_2),$$
  $$\mathcal{L}(F_1 ; F_2) = \mathcal{L}(F_1) \cdot \mathcal{L}(F_2).$$
Lemma 3.4. For all RDC formulae $F$, discrete interpretations $\mathcal{I}$, $n \geq 0$, and all words $w \in \Sigma(F)^*$ which describe $\mathcal{I}$ on $[0, n]$, 

$\mathcal{I}, [0, n] \models F$ if and only if $w \in \mathcal{L}(F)$.

Proof: Structural induction

Base $\not\models \Gamma P$: Assume $u = a_1 \ldots a_n$ describes $\mathcal{I}$ on $[0,n]$

$\mathcal{I}, [0,n] \models \Gamma P$ $\iff$ $\mathcal{I}, [0,n] \models \Gamma P$ and $n \geq 1$

$\iff$ $n \geq 1$ and $\forall 1 \leq j \leq n \mathcal{I}, [j-1,j] \models \Gamma P$

$\iff$ $n \geq 1$ and $\forall 1 \leq j \leq n \mathcal{I}, [j-1,j] \models (\Gamma P \lor e_j)$ and $e_j \in \text{DNF}(\Gamma P)$

$\iff$ $\forall 1 \leq j \leq n \mathcal{I}, [j-1,j] \models \text{DNF}(\Gamma P)$ and $e_j \in \text{DNF}(\Gamma P)$ $\cup$ clean

$\iff$ $w \in \text{DNF}(\Gamma P)$

$\iff$ $w \in \mathcal{L}(\Gamma P)$. 

Steps:

1. $\not\models \Gamma P$

2. $\not\models \Gamma P \lor e_2$

3. $\not\models \Gamma P \lor e_3$
Sketch: Proof of Theorem 3.9

Theorem 3.9.
The realisability problem for RDC with discrete time is decidable.

- $kern(L)$ contains all words of $L$ whose prefixes are again in $L$.
- If $L$ is regular, then $kern(L)$ is also regular.
- $kern(L(F))$ can effectively be constructed.
- We have

Lemma 3.8. For all RDC formulae $F$, $F$ is realisable from 0 in
discrete time if and only if $kern(L(F))$ is infinite.

- Infinity of regular languages is decidable.
(Variants of) RDC in Continuous Time
Recall: Restricted DC (RDC)

\[ F ::= [P] | \neg F_1 | F_1 \lor F_2 | F_1 ; F_2 \]

where \( P \) is a state assertion, but with \textbf{boolean} observables only.

From now on: “RDC + \( \ell = x, \forall x \)”

\[ F ::= [P] | \neg F_1 | F_1 \lor F_2 | F_1 ; F_2 | \ell = 1 | \ell = x | \forall x \bullet F_1 \]
Theorem 3.10.
The realisability from 0 problem for DC with continuous time is undecidable, not even semi-decidable.

Theorem 3.11.
The satisfiability problem for DC with continuous time is undecidable.
Sketch: Proof of Theorem 3.10

Reduce divergence of two-counter machines to realisability from 0:

- Given a two-counter machine $M$ with final state $q_{\text{fin}}$,
- construct a DC formula $F(M) := encoding(M)$
- such that
  \[ M \text{ diverges if and only if } F(M) \land \neg \Diamond [q_{\text{fin}}] \]
  is realisable from 0.

- If realisability from 0 was (semi-)decidable,
  divergence of two-counter machines would be (which it isn’t).
Recall: Two-counter machines

A two-counter machine is a structure

\[ M = (Q, q_0, q_{\text{fin}}, \text{Prog}) \]

where

- \( Q \) is a finite set of states,
- comprising the initial state \( q_0 \) and the final state \( q_{\text{fin}} \)
- \( \text{Prog} \) is the machine program, i.e. a finite set of commands of the form
  
  \[ q : \text{inc}_i : q' \quad \text{and} \quad q : \text{dec}_i : q', q'', \quad i \in \{1, 2\}. \]

- We assume deterministic 2CM: for each \( q \in Q \), at most one command starts in \( q \), and \( q_{\text{fin}} \) is the only state where no command starts.
2CM Configurations and Computations

- A **configuration** of $M$ is a triple $K = (q, n_1, n_2) \in Q \times \mathbb{N}_0 \times \mathbb{N}_0$.

- The **transition relation** $\vdash$ on configurations is defined as follows:

<table>
<thead>
<tr>
<th>Command</th>
<th>Semantics: $K \vdash K'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q : inc_1 : q'$</td>
<td>$(q, n_1, n_2) \vdash (q', n_1 + 1, n_2)$</td>
</tr>
<tr>
<td>$q : dec_1 : q', q''$</td>
<td>$(q, 0, n_2) \vdash (q', 0, n_2)$, $(q, n_1 + 1, n_2) \vdash (q'', n_1, n_2)$</td>
</tr>
<tr>
<td>$q : inc_2 : q'$</td>
<td>$(q, n_1, n_2) \vdash (q', n_1, n_2 + 1)$</td>
</tr>
<tr>
<td>$q : dec_2 : q', q''$</td>
<td>$(q, n_1, 0) \vdash (q', n_1, 0)$, $(q, n_1, n_2 + 1) \vdash (q'', n_1, n_2)$</td>
</tr>
</tbody>
</table>

- The (!) **computation** of $M$ is a finite sequence of the form $\langle \ldots \rangle$ (“$M$ **halts**”) or an infinite sequence of the form $\langle \ldots \rangle$ (“$M$ **diverges**”)
2CM Example

- \( M = (Q, q_0, q_{\text{fin}}, \text{Prog}) \)
- commands of the form \( q : inc_i : q', q'' \) and \( q : dec_i : q', q'' \), \( i \in \{1, 2\} \)
- configuration \( K = (q, n_1, n_2) \in Q \times \mathbb{N}_0 \times \mathbb{N}_0 \).

<table>
<thead>
<tr>
<th>Command</th>
<th>Semantics: ( K \vdash K' )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q : inc_1 : q' )</td>
<td>( (q, n_1, n_2) \vdash (q', n_1 + 1, n_2) )</td>
</tr>
<tr>
<td>( q : dec_1 : q', q'' )</td>
<td>( (q, 0, n_2) \vdash (q', 0, n_2) ), ( (q, n_1 + 1, n_2) \vdash (q'', n_1, n_2) )</td>
</tr>
<tr>
<td>( q : inc_2 : q' )</td>
<td>( (q, n_1, n_2) \vdash (q', n_1, n_2 + 1) )</td>
</tr>
<tr>
<td>( q : dec_2 : q', q'' )</td>
<td>( (q, n_1, 0) \vdash (q', n_1, 0) ), ( (q, n_1, n_2 + 1) \vdash (q'', n_1, n_2) )</td>
</tr>
</tbody>
</table>
Reducing Divergence to DC realisability: Idea In Pictures

2CN M diverges if

exists \pi: K_0 \rightarrow k_1 \rightarrow k_2 \ldots

iff

exist

I

\sim K_0 \sim K_1 \sim K_2 \ldots

("I describes \pi")

and

I K_0 F(M) \sim 0 [k_n I]

F(M) intuitively requires:

- \[0,d] encodes \(g_0, 0, 0\)
- \[n.d, (n+1).d\] encodes d configuration
- \[n.d, (n+1).d\] and \[(n+1).d, (n+2).d\]
  encode configurations which are \(m^1\) - relation

- if \(q_{fin}\) is reached, we stay there
Reducing Divergence to DC realisability: Idea

- A single configuration $K$ of $\mathcal{M}$ can be encoded in an interval of length 4; being an encoding interval can be characterised by a DC formula.

- An interpretation on ‘Time’ encodes the computation of $\mathcal{M}$ if
  - each interval $[4n, 4(n + 1)]$, $n \in \mathbb{N}_0$, encodes a configuration $K_n$,
  - each two subsequent intervals $[4n, 4(n + 1)]$ and $[4(n + 1), 4(n + 2)]$, $n \in \mathbb{N}_0$, encode configurations $K_n \vdash K_{n+1}$ in transition relation.

- Being encoding of the run can be characterised by DC formula $F(\mathcal{M})$.

- Then $\mathcal{M}$ diverges if and only if $F(\mathcal{M}) \land \neg \lozenge [q_{\text{fin}}]$ is realisable from 0.
Encoding Configurations

- We use $\text{Obs} = \{\text{obs}\}$ with $\mathcal{D}(\text{obs}) = Q_M \cup \{C_1, C_2, B, X\}$.

Examples:

- $K = (q, 2, 3)$

$$
\left(\begin{array}{c}
[q] \\
\wedge
\end{array}\right); \left(\begin{array}{c}
[B]; [C_1]; [B]; [C_1]; [B] \\
\wedge \\
\ell = 1
\end{array}\right); \left(\begin{array}{c}
[X] \\
\wedge \\
\ell = 1
\end{array}\right); \left(\begin{array}{c}
[B]; [C_2]; [B]; [C_2]; [B]; [C_2]; [B] \\
\wedge \\
\ell = 1
\end{array}\right)
$$

- $K_0 = (q_0, 0, 0)$

$$
\left(\begin{array}{c}
[q_0] \\
\wedge \\
\ell = 1
\end{array}\right); \left(\begin{array}{c}
[B] \\
\wedge \\
\ell = 1
\end{array}\right); \left(\begin{array}{c}
[X] \\
\wedge \\
\ell = 1
\end{array}\right); \left(\begin{array}{c}
[B] \\
\wedge \\
\ell = 1
\end{array}\right)
$$

or, using abbreviations, $[q_0]^1; [B]^1; [X]^1; [B]^1$. 
Construction of $F(\mathcal{M})$

In the following, we give DC formulae describing

- the initial configuration,
- the general form of configurations,
- the transitions between configurations,
- the handling of the final state.

$F(\mathcal{M})$ is the conjunction of all these formulae.

$$F(\mathcal{M}) = \text{init} \land \text{keep} \land \ldots$$

$$\land \land \land \quad F(q = \text{inc}; q')$$

$q : \text{inc} \land q' \in \text{Pos}(\mathcal{M})$

$$\land \land \land \quad F(q = \text{dec}; q')$$

$q : \text{dec} \land q' \in \text{Pos}(\mathcal{M})$
Initial and General Configurations

\[
\text{init} \iff (\ell \geq 4 \implies [q_0]^{1}; [B]^{1}; [X]^{1}; [B]^{1}; \text{true})
\]

\[
\text{keep} \iff \square([Q]^{1}; [B \lor C_1]^{1}; [X]^{1}; [B \lor C_2]^{1}; \ell = 4 \\
\implies \ell = 4; [Q]^{1}; [B \lor C_1]^{1}; [X]^{1}; [B \lor C_2]^{1})
\]

where \( Q := \neg (X \lor C_1 \lor C_2 \lor B) \).
$$\text{Auxiliary Formula Pattern copy}$$

$$\text{copy}(F, \{P_1, \ldots, P_n\}) : \Longleftrightarrow$$

$$\forall c, d \bullet \Box((F \land \ell = c) ; ([P_1 \lor \cdots \lor P_n] \land \ell = d) ; [P_1] ; \ell = 4$$

$$\Rightarrow \ell = c + d + 4 ; [P_1]$$

$$\wedge \ldots$$

$$\forall c, d \bullet \Box((F \land \ell = c) ; ([P_1 \lor \cdots \lor P_n] \land \ell = d) ; [P_n] ; \ell = 4$$

$$\Rightarrow \ell = c + d + 4 ; [P_n]$$
(i) Change state
\[ \Box([q]^1; [B \lor C_1]^1; [X]^1; [B \lor C_2]^1; \ell = 4 \implies \ell = 4; [q']^1; \text{true}) \]

(ii) Increment counter
\[ \forall d \cdot \Box([q]^1; [B]^d; (\ell = 0 \lor [C_1]; [\neg X]); [X]^1; [B \lor C_2]^1; \ell = 4 \implies \ell = 4; [q']^1; ([B]; [C_1]; [B]) \land \ell = d); \text{true} \]
\( q : \text{inc}_1 : q' \) (Increment)

(i) Keep rest of first counter

\[
\text{copy}(\lceil q \rceil^1 ; [B \lor C_1] ; [C_1], \{B, C_1\})
\]

(ii) Leave second counter unchanged

\[
\text{copy}(\lceil q \rceil^1 ; [B \lor C_1] ; [X]^1, \{B, C_2\})
\]
\( q : \text{dec}_1 : q', q'' \ (\text{Decrement}) \)

(i) If zero

\[
\square([q]^1 ; [B]^1 ; [X]^1 ; [B \lor C_2]^1 ; \ell = 4 \implies \ell = 4 ; [q']^1 ; [B]^1 ; \text{true})
\]

(ii) Decrement counter

\[
\forall d \bullet \square([q]^1 ; ([B] ; [C_1] \land \ell = d) ; [B] ; [B \lor C_1] ; [X]^1 ; [B \lor C_2]^1 ; \ell = 4 \implies \ell = 4 ; [q'']^1 ; [B]^d ; \text{true})
\]

(iii) Keep rest of first counter

\[
copy([q]^1 ; [B] ; [C_1] ; [B_1], \{B, C_1\})
\]

(iv) Leave second counter unchanged

\[
copy([q]^1 ; [B \lor C_1] ; [X]^1, \{B, C_2\})
\]
Final State

\[ copy([q_{\text{fin}}]^1; [B \lor C_1]^1; [X]; [B \lor C_2]^1, \{q_{\text{fin}}, B, X, C_1, C_2\}) \]
Following [Chaochen and Hansen, 2004] we can observe that

\[ \mathcal{M} \text{ halts if and only if } \text{the DC formula } F(\mathcal{M}) \land \lozenge \lceil q_{\text{fin}} \rceil \text{ is satisfiable.} \]

This yields

**Theorem 3.11.** The satisfiability problem for DC with continuous time is undecidable.

(It is semi-decidable.)

Furthermore, by taking the contraposition, we see

\[ \mathcal{M} \text{ diverges if and only if } \mathcal{M} \text{ does not halt if and only if } F(\mathcal{M}) \land \neg \lozenge \lceil q_{\text{fin}} \rceil \text{ is not satisfiable.} \]

Thus whether a DC formula is **not satisfiable** is not decidable, not even semi-decidable.
Validity

- By Remark 2.13, $F$ is valid iff $\neg F$ is not satisfiable, so

**Corollary 3.12.** The validity problem for DC with continuous time is undecidable, not even semi-decidable.

- This provides us with an alternative proof of Theorem 2.23 ("there is no sound and complete proof system for DC"):  

  - **Suppose** there were such a calculus $C$.
  - By Lemma 2.22 it is semi-decidable whether a given DC formula $F$ is a theorem in $C$.
  - By the soundness and completeness of $C$, $F$ is a theorem in $C$ if and only if $F$ is valid.
  - Thus it is semi-decidable whether $F$ is valid. **Contradiction.**
Discussion

• Note: the DC fragment defined by the following grammar is **sufficient** for the reduction

\[ F ::= \left[ P \right] \mid \neg F_1 \mid F_1 \lor F_2 \mid F_1 ; F_2 \mid \ell = 1 \mid \ell = x \mid \forall x \bullet F_1, \]

\( P \) a state assertion, \( x \) a global variable.

• Formulae used in the reduction are abbreviations:

\[
\ell = 4 \iff \ell = 1 ; \ell = 1 ; \ell = 1 ; \ell = 1 \\
\ell \geq 4 \iff \ell = 4 ; \text{true} \\
\ell = x + y + 4 \iff \ell = x ; \ell = y ; \ell = 4
\]

• Length 1 is not necessary — we can use \( \ell = z \) instead, with fresh \( z \).

• This is RDC augmented by "\( \ell = x \)" and "\( \forall x \)",
which we denote by **RDC + \( \ell = x, \forall x \)**.
References
References
